# A BDD Proposal for Probabilistic Switching Activity Estimation

*Felipe Machado*[1], *Yago Torroja*[2], *Teresa Riesgo*[2]
[1]*Departamento de Tecnología Electrónica, ESCET, Universidad Rey Juan Carlos*
*c/ Tulipán s/n. 28933 Móstoles, Madrid, Spain*
[2] *Centro de Electrónica Industrial, ETSII, Universidad Politécnica de Madrid*
*c/ José Gutiérrez Abascal, 2. 28006 Madrid, Spain*
*felipe.machado@urjc.es, {yago.torroja, teresa.riesgo}@upm.es*

***Abstract*— Switching activity computation is a essential stage for dynamic power estimation. Binary decision diagrams (BDD) are widely used in probabilistic activity estimation. However, the BDD size used for switching activity increases significantly in respect to the logic function BDDs.**

**In this paper we propose a new BDD structure for activity computations in which important size reductions are achieved with no accuracy loss. The proposal includes the definition of a BDD activity operator. This operator has been implemented in a BDD package and then, in an automated tool. This implementation has permitted the analysis of several circuits and has corroborated the size reductions and the accuracy of the results.**

***Index Terms*— Switching activity, power estimation, BDD, binary decision diagram**

## I. INTRODUCTION

POWER consumption has become a critical issue in digital circuits due to the extreme power densities of high performance circuits and due to the widespread usage of mobile devices. Therefore, power consumption has to be considered since the beginning of a circuit design and hence, design tools are needed in order to reduce the power consumption of the circuit [1]. Among these tools, estimation tools are necessary in order to anticipate the final characteristics of the circuits. This anticipation will favor a seamless design flow, helping to avoid the costly design iterations.

One of the main sources of power dissipation is the dynamic power consumption, which is caused by the transitions between the logic levels of the internal signals. The average of these transitions is known as switching activity or just activity.

Nevertheless, estimating a circuit's switching activity is a hard task because it not only depends on the circuit structure and functionality, but it also depends on the inputs behavior. To address this problem, two methods have been proposed at gate level: dynamic and static methods.

Dynamic methods simulate the circuit and then, the activity statistics of the simulation are extracted. These methods are easy to implement, but the results depend on the input vector used and, what it worst, the simulation times could be so large that the method would become unfeasible.

Static methods commonly use probabilistic techniques to propagate the switching activity from input ports to outputs. These methods are faster but their main disadvantage is the complexity and size of the probabilistic models. In particular, these methods typically use ROBDDs (reduced ordered binary decision diagrams [2], in short, just BDDs henceforth) whose size may extraordinarily increase with the circuit size. The problem is aggravated when BDDs are used to compute the switching activity because their size increases even more.

In this paper we propose a BDD structure for probabilistic switching activity estimation. This structure reduces the number of nodes of the BDDs from 25% to 50% without any accuracy loss in the results. This work is an extension of a previous work [3], giving further details of the implementation and results. This proposal complements the ongoing work aiming to reduce the complexity of the probabilistic activity estimation [12], [13].

This paper is structured as follows: in the next section the previous work is expounded. Then, section III explains the proposed activity BDDs. Afterwards, experimental results are shown. To end, the conclusions of this work are given.

## II. PREVIOUS WORK

Many proposals have contributed to the probabilistic estimation of the switching activity. The first proposals calculated the activity by means of the probability. In this sense, Cirit [4] calculated the activity using the equation:

$$a_x^{ti} = 2\,P_x\,\overline{P}_x = 2\,P_x\,(1-P_x)\,. \qquad (1)$$

This equation assumes temporal independence (*ti*); hence, switching activity can be calculated directly from signal probability.

Nevertheless, equation (1) produces inaccurate results. In order to avoid these inaccuracies, Najm [5] proposed an algorithm to propagate probabilities and transition densities using BDDs. The transition density (*D*) was formulated as:

$$D(f) = \sum_{i=1}^{n} P\left(\frac{\partial f}{\partial x_i}\right) \cdot D(x_i)\ . \qquad (2)$$

Where $f$ is a function depending on $x_i$, and $\partial f/\partial x_i$ is the boolean difference:

$$\frac{\partial f}{\partial x_i} = f\big|_{x_i=1} \oplus f\big|_{x_i=0} = f(x_i) \oplus f(\overline{x}_i)\ . \qquad (3)$$

Where symbol $\oplus$ represents the exclusive-OR operator.

However, equation (2) does not consider the effect of simultaneous transitions of the inputs. Thus, Ghosh [6] proposed to apply the XOR of the output function $f$ at two consecutive times:

$$f^0 \oplus f^T = f(t{=}0) \oplus f(t{=}T) = (f^0 \wedge \neg f^T) \vee (\neg f^0 \wedge f^T)\,. \quad (4)$$

With this solution, simultaneous transitions of the inputs are considered. Taking the probability of (4), the probability of a transition at two consecutive time steps is obtained, which is the definition of the switching activity when

spurious transitions (glitches) are not considered:

$$a_f = P\{f^0 \oplus f^T\}\ . \qquad (5)$$

In this work, BDDs were used in order to perform the computations.

In a similar way, Schneider [7] calculated the activity evaluating the probability for the output function to change its value: $Tf = f^0 \oplus f^T$, what was called *transition function*. The formalisms to represent $Tf$ using BDDs were developed in this work. This BDD representation was named TFBDD. The activity was calculated using the Shannon expansion of the probability of $Tf$:

$$a = P(Tf) = P\{(A^0 \wedge Tf_{A(0)}) \vee (\neg A^0 \wedge Tf_{\overline{A}(0)})\} =$$

$$= P\Big\{(A^0 \wedge A^T \wedge Tf_{A(0)A(T)}) \vee (A^0 \wedge \neg A^T \wedge Tf_{A(0)\overline{A}(T)}) \vee$$

$$\vee (\neg A^0 \wedge A^T \wedge Tf_{\overline{A}(0)A(T)}) \vee (\neg A^0 \wedge \neg A^T \wedge Tf_{\overline{A}(0)\overline{A}(T)})\Big\}\ . \qquad (6)$$

Where $A^0$ and $A^T$ represent input $A$ at times 0 and $T$ respectively. $Tf_{A(0)}$ is the cofactor of $Tf$ in respect to $A^0$ and $Tf_{A(0)A(T)}$ is the cofactor of $Tf_{A(0)}$ in respect to $A^T$.

Since $\{A^0 \wedge A^T\}$; $\{A^0 \wedge \neg A^T\}$; $\{\neg A^0 \wedge A^T\}$; and $\{\neg A^0 \wedge \neg A^T\}$ are mutually disjoint events, considering that the inputs are mutually independent and since each cofactor does not depend on $A$, equation (6) results in:

$$a = P(A_{1\to1})\cdot P\left(Tf_{A(0)A(T)}\right) + P(A_{1\to0})\cdot P\left(Tf_{A(0)\overline{A}(T)}\right) +$$

$$+ P(A_{0\to1})\cdot P\left(Tf_{\overline{A}(0)A(T)}\right) + P(A_{0\to0})\cdot P\left(Tf_{\overline{A}(0)\overline{A}(T)}\right)\ . \qquad (7)$$

Equation (7) separates input $A$ from the transition function. All the other input signals are then also separated performing the same operation recursively.

To represent (7) in a BDD, the TFBDD has to be orderered in such a way that each signal $A$ at time 0 has to be followed by itself at time $T$. Equation (7) is represented in figure 1.

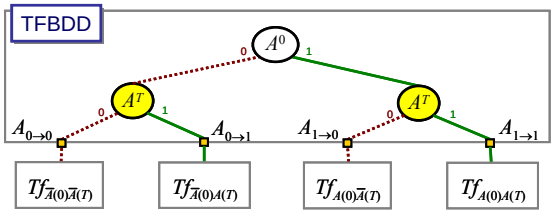


Fig. 1: TFBDD representation of (7)

Nevertheless, some cofactors could be identical; therefore the general structure of figure 1 would be simplified. For example, when cofactors are equal: $Tf_{A(0)A(T)} = Tf_{A(0)\overline{A}(T)}$, they are equal to $Tf_{A(0)}$, and equation (7) would be

$$a = \left(P(A_{1\to1}) + P(A_{1\to0})\right)\cdot P\left(Tf_{A(0)}\right) +$$

$$+ P(A_{0\to1})\cdot P\left(Tf_{\overline{A}(0)A(T)}\right) + P(A_{0\to0})\cdot P\left(Tf_{\overline{A}(0)\overline{A}(T)}\right)\ .$$

Since $P(A^0) = P(A(t=0)) = P(A_{1\to1}) + P(A_{1\to0})$, then:

$$a = P(A^0)\cdot P\left(Tf_{A(0)}\right) +$$

$$+ P(A_{0\to1})\cdot P\left(Tf_{\overline{A}(0)A(T)}\right) + P(A_{0\to0})\cdot P\left(Tf_{\overline{A}(0)\overline{A}(T)}\right)\ . \qquad (8)$$

Whose TFBDD would be represented as

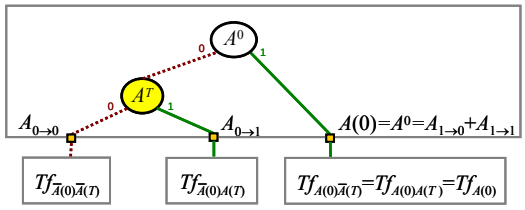


Fig. 2: TFBDD representation when $Tf_{A(0)A(T)} = Tf_{A(0)\overline{A}(T)}$. Equation (8)

Other typical simplifications happens when $Tf_{A(0)A(T)} = Tf_{A(0)\overline{A}(T)}$ and $Tf_{\overline{A}(0)A(T)} = Tf_{\overline{A}(0)\overline{A}(T)}$, (fig 3.a). Or when $Tf_{A(0)} = Tf_{\overline{A}(0)}$, (fig. 3.b).

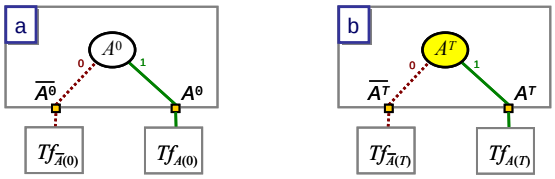


Fig. 3: TFBDD representations when a) $Tf_{A(0)A(T)} = Tf_{A(0)\overline{A}(T)}$ and $Tf_{\overline{A}(0)A(T)} = Tf_{\overline{A}(0)\overline{A}(T)}$. And when b) $Tf_{A(0)} = Tf_{\overline{A}(0)}$

The process of building a TFBDD consists in performing the XOR of the output logic function BDD (or probability BDD) at two consecutive times. This process is shown in figure 4.

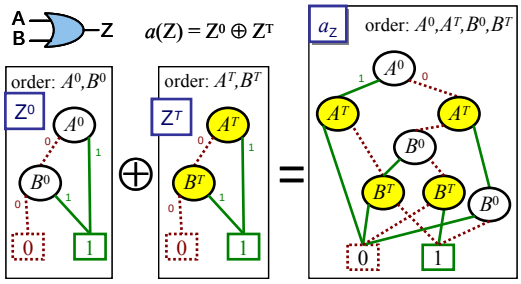


Fig. 4: TFBDD creation of an OR from the output function

These are the fundamentals of TFBDDs proposed by Schneider [7]. They have been explained in such detail because our work proposes a more efficient representation for the activity BDDs.

In a later work, Marculescu [8] did not apply the XOR operator to the probability BDD. Instead he directly used probability BDDs and calculated transition probabilities through a dynamic programming approach. He claimed that his method is more efficient than applying the XOR; however, BDD operations can also be implemented through dynamic approaches [9] and there are BDD packages which implements them [10]. We consider that using these packages the complexity is similar for both cases.

Many other proposals also use BDDs. For example, Theodoridis [11], implemented a more sophisticated version of activity BDDs in which spurious transitions are also accounted. Adopting the concept of *timed Boolean functions* (TBF), the proposed BDD are called TBF-OBDD. Obviously, the complexity of these BDDs increases considerably.

## III. Proposed Activity BDDs

With the purpose of reducing the activity BDD size, we propose a new approach of representing them whereby significant BDD size reductions are achieved.

Instead of using a timed based representation ($x^0$ and $x^T$) as TFBDDs do, we propose a representation based firstly on activity and inactivity ($a_x$ and $\overline{a}_x$); and then on transition probability: $P(x_{0\to0})$, $P(x_{1\to1})$, $P(x_{0\to1})$, $P(x_{1\to0})$. This kind of representation has been called activity BDD, or in its short form *a*BDD.

This section is organized as follows: first the *a*BDD structure will be explained. Then, the structures of TFBDDs and *a*BDD are compared. Afterwards, an activity operator is

defined in order to build the proposed *a*BDDs. Last, the size of TFBDDs and *a*BDDs of some logic gates and multiplexers are compared.

### *A. Structure of the proposed activity BDD*

The general *a*BDD structure for any signal $A$ is shown in figure 5. Instead of creating a new variable at time $T$ (i.e. $A^T$) as TFBDDs do (fig. 1) a new activity variable ($a_A$) is created. In *a*BDDs, the variable order of different signals is maintained, but the new activity variables are set immediately before the variables of their same signal.

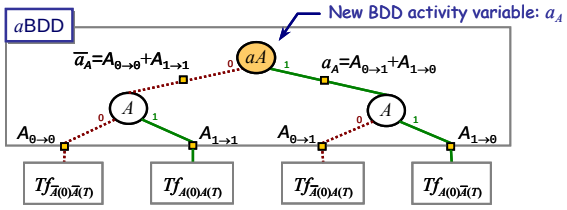


Fig. 5: *a*BDD representation of (7)

From figure 5 it can be seen that, the edge zero of the activity node represents the signal inactivity, $\overline{a}_A$ :

$$\overline{a}_A = P(A_{0\to0}) + P(A_{1\to1}) \quad (9)$$

The other edge (edge 1) represents the signal activity, $a_A$ :

$$a_A = P(A_{0\to1}) + P(A_{1\to0}) \quad (10)$$

When a signal node ($A$) follows its activity node ($a_A$), the edges of the signal node separate the above formulas in their terms. For example, if the edge zero of the activity node is taken, it represents the signal inactivity: $\overline{a}_A = P(A_{0\to0}) + P(A_{1\to1})$. Then, from the signal node $A$, if edge 0 is taken, it means that the signal has no switching activity and it remains at zero. Therefore, this path represents the transition: $A_{0\to0}$. If, instead of taking this last edge, edge 1 were taken, it would also mean that signal $A$ does not have activity, but the signal remains at one: $A_{1\to1}$.

On the contrary, if edge 1 of the activity node is taken, it represents the signal activity $a_A = P(A_{0\to1}) + P(A_{1\to0})$. Then, taking edge 0 of the signal node it represents the transition $A_{0\to1}$. And the other edge represents the transition $A_{1\to0}$. However, since $P(A_{0\to1})$ and $P(A_{1\to0})$ are equal: $P(A_{0\to1}) = P(A_{1\to0})$, their corresponding cofactors may be interchanged in the structure.

Similar to TFBDDs, usually *a*BDD structures can also be simplified. Figure 6.a shows a common situation in which the activity cofactors are equal: $Tf_{\overline{A}(0)A(T)} = Tf_{A(0)\overline{A}(T)}$, what has been called $Tf_{aA}$. When the inactivity cofactors are also equal, the resulting *a*BDD structure is shown in figure 6.b. In this case $Tf_{A(0)A(T)} = Tf_{\overline{A}(0)\overline{A}(T)}$. What has been named as $Tf_{\overline{a}A}$.

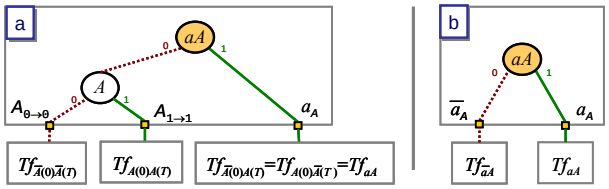


Fig. 6: a) *a*BDD representation when $Tf_{\overline{A}(0)A(T)} = Tf_{A(0)\overline{A}(T)} (= Tf_{aA})$
b) when $Tf_{\overline{A}(0)A(T)} = Tf_{A(0)\overline{A}(T)}$ and $Tf_{\overline{A}(0)\overline{A}(T)} = Tf_{A(0)A(T)} (= Tf_{\overline{a}A})$

None of the operations have any implication in the accuracy; therefore the activity results will be equivalent to those obtained from TFBDDs.

### *B. Comparison between TFBDD and aBDD structures*

Due to the structural differences between *a*BDDs and TFBDDs, *a*BDDs achieve a considerable node reduction compared to TFBDDs. There is no difference in the number of nodes between the general structures of *a*BDDs and TFBDDs (figures 1 and 5). However, many of the *a*BDD simplified structures are more advantageous than the TFBDD structures.

Figure 7 compares the structures when the activity cofactors are equal (fig. 6.a), which is a very common situation. In the TFBDD (on the left), both paths related to transition 0→1 and transition 1→0 finish at the same node. The union of both transitions represents the switching activity, which precisely is the meaning of edge 1 of the *a*BDD activity node. As a consequence, one node out of three is saved using *a*BDDs.

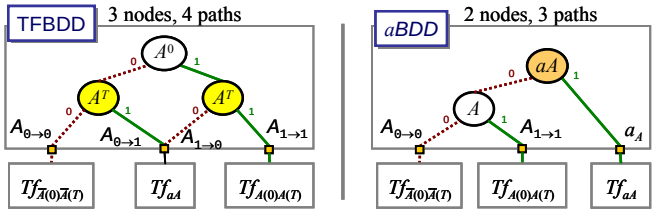


Fig. 7: Comparison between TFBDD and aBDD structures when the activity cofactors are equal.

Another even more advantageous situation happens when inactivity cofactors are also equal (fig. 6.b). Transitions 0→0 and 1→1 end in the same node, but different to the node in which transitions 0→1 and 1→0 finish. Transitions 0→0 and 1→1 represent the signal inactivity, which is directly obtained by the edge zero of the activity node. Figure 8 shows the difference between the TFBDD and the *a*BDD structures. In this case, two nodes out of three have been saved and two paths out of four.

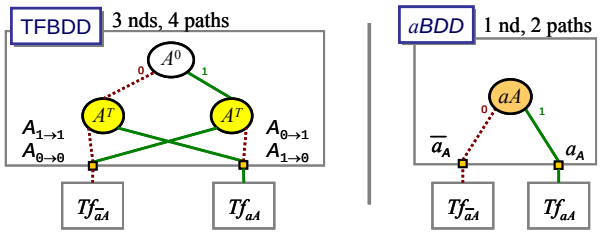


Fig. 8: Comparison between TFBDD and *a*BDD structures when both activity and inactivity cofactors are equal

To end, figure 9 shows how signal probability is represented in TFBDDs and *a*BDDs. If TFBDDs are used, two representations are possible: one for the probability at time 0 and another at time $T$. However, these probabilities are equal and they can be substituted by the signal probability without specifying time. That is exactly what *a*BDDs do, hence, a node can be saved when both structures are used in TFBDDs.

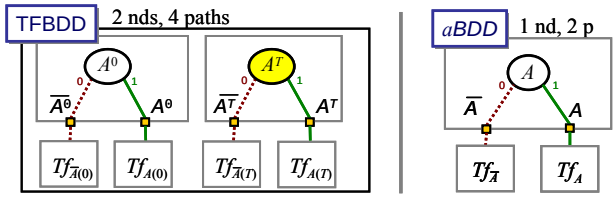


Fig. 9: Comparison between the TFBDDs and *a*BDDs used to represent signal probability

There are more situations in which reductions are achieved; however, they will not be illustrated since they are variations of the already described cases

### C. *Definition of an activity operator*

The transformation from TFBDDs into *a*BDDs has been outlined in a previous work [3]. However, it is not very useful to propose an optimized BDD representation (*a*BDD) if the previous representation (TFBDD) is needed to obtain the proposed one. Therefore, in this section the activity operator definition is exposed so as to create *a*BDDs directly from the probability BDDs, without using TFBDDs.

There are operators defined for BDDs, for example: AND, XOR, composition,... Since a new BDD representation has been defined, an operator to build it has to be defined. Once the operator is defined, it can be implemented as a function in a library package for manipulating BDDs.

The first step to create the *a*BDD is the variable ordering definition. The *a*BDD order is the same as the probability BDD order, but the signal activity variables are just before their corresponding signal variables. That is, if the BDD variable order is $\{A, B, C\}$ the *a*BDD variable order is $\{a_A, A, a_B, B, a_C, C\}$.

Only the original probability BDD is needed to create the *a*BDD. It is important to remark that the activity operator is defined for just one operand but, because it is a recursive function, when traversing the BDD, in the subsequent calls to the function it may have different operands. Therefore, it is no sense to apply the activity operator to different operands, but it makes sense to call the function with different arguments when traversing the BDD.

Consequently, the activity operator is defined for two operands, which are probability BDDs: $\mathcal{M}$ and $\mathcal{N}$ (probability BDDs will be written in this kind of cursive font). The result of the operator is an activity BDD: $a\mathcal{Q}$ (activity BDDs will be written in the same kind of cursive font, but they will be preceded by letter "*a*"). Therefore, the transformation is:

$$a[\mathcal{M},\mathcal{N}] \rightarrow a\mathcal{Q} \tag{11}$$

Therefore, the domain of the function is formed by all the probability BDDs and the image (or range) of the function are the *a*BDDs.

When the function only has one operand ($\mathcal{M}$), the operation is carried out over the same operand:

$$a[\mathcal{M}] = a[\mathcal{M},\mathcal{M}] \tag{12}$$

Since it is not easy to use formulas to represent BDDs, in order to facilitate the comprehension, figures representing the operations will also be used. Figure 10 represents equation 12. The operands are between squared brackets and the operands are probability BDDs.

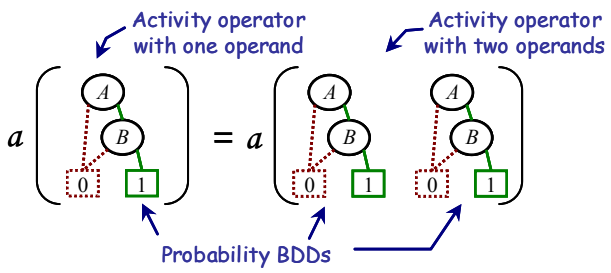


Fig. 10: Activity operator equivalence for one operand

The activity operator is commutative:

$$a[\mathcal{M},\mathcal{N}] = a[\mathcal{M},\mathcal{N}] \tag{13}$$

It is no sense to analyze the associative and distributive properties of the activity operator because the domain of the function (probability BDDs) is different than the image (*a*BDD). Therefore, the result of the operation cannot be operated again.

The activity operator applied to a constant BDD (*0* or *1*) is the *a*BDD zero:

$$a[0] = a[1] = 0 \tag{14}$$

Then also:

$$a[0,0] = a[1,1] = 0 \tag{15}$$

Equations 14 and 15 are represented in figure 11:

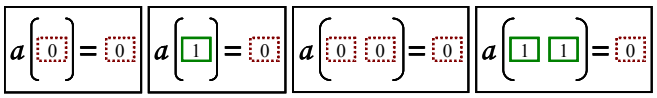

Fig. 11: Activity operator applied to constant and equal BDDs

The activity equation applied to constant but different BDDs equals the *a*BDD one:

$$a[0,1] = a[1,0] = 1 \tag{16}$$

Equation 16 can be represented as shown in figure 12.

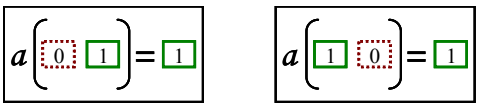

Fig. 12: Activity function applied to constant but different BDDs.

BDD one and BDD zero belongs to both the domain (probability BDD) and the image (*a*BDD).

A simple node is a node whose edges end in terminal nodes but different nodes. That is, one edge leads to terminal node 1 and the other leads to terminal node 0. The activity operator applied to a simple node results in the activity node of the signal of the original node:

$$a[\mathcal{A}] = a\mathcal{A} \tag{17}$$

Being $\mathcal{A}$ a simple probability BDD of signal $A$, and $a\mathcal{A}$ the *a*BDD of signal $A$. To clarify it, figure 13 shows the representation of equation 17.

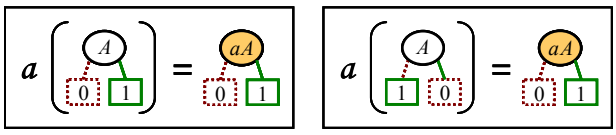

Fig. 13: Activity operator applied to simple probability BDDs

The activity operator applied to simple nodes of variable $A$, but being one the negated, results in the inactivity of $A$.

$$a[\mathcal{A}, \overline{\mathcal{A}}] = \overline{a\mathcal{A}} \tag{18}$$

Where $\overline{\mathcal{A}}$ is a simple probability BDD that is the negated form of $\mathcal{A}$. That is, it represents $\overline{P}_A$. While $\overline{a\mathcal{A}}$ is the *a*BDD that represents the inactivity of signal $A$. Equation 18 has been represented in figure 14.

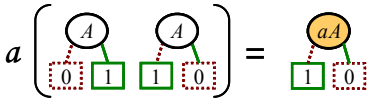

Fig. 14: Activity function applied to simple probability BDDs of the same signal but one of them negated

The activity function applied to a BDD and the terminal node zero results in the same BDD:

$$a[\mathcal{M}, 0] = \mathcal{M} \tag{19}$$

The activity function applied to a BDD and the terminal

node one results in the negated BDD:

$$a[\mathcal{M}, 1] = \overline{\mathcal{M}} \quad (20)$$

These two properties embrace equations 15 and 16; as a consequence, terminal node zero can be considered as the neutral element of the activity operator.

The activity function applied to BDDs with different variables in their root nodes is executed in a recurrent way through the child nodes. Let $\mathcal{M}$ and $\mathcal{N}$ be the operands, and assuming that the variable of the root node of $\mathcal{M}$ precedes the variable of the root node of $\mathcal{N}$. Then:

- The *a*BDD root node is created with the same variable as the root node of $\mathcal{M}$ (the variable that goes first in the BDD ordering)
- Attached to edge zero of the *a*BDD root node is the result of the activity function applied to $\mathcal{N}$ and the cofactor of $\mathcal{M}$ in respect to the negated variable of the root node
- Attached to edge one of the *a*BDD root node is the result of the activity function applied to $\mathcal{N}$ and the cofactor of $\mathcal{M}$ in respect to the variable of the root node

What can be formulated in the following equation:

$$a[\mathcal{M}, \mathcal{N}]=\text{ITE}\left\{\text{root}(\mathcal{M}), a[\mathcal{M}_{\text{root}(\mathcal{M})},\mathcal{N}], a[\mathcal{M}_{\overline{\text{root}(\mathcal{M})}},\mathcal{N}]\right\} \quad (21)$$

Where ITE($A$, $\mathcal{R}$, $\mathcal{S}$) is the function *if-then-else* for BDDs. The first argument $A$ of the function ITE is a BDD variable, while the second and third arguments ($\mathcal{R}$ and $\mathcal{S}$) are BDDs. Function ITE builds a BDD creating the root node containing the variable of the first argument ($A$). Then, the second argument ($\mathcal{R}$) is attached to the edge 1 of the root node. And the third argument ($\mathcal{S}$) is attached to the edge 0 of the root node.

Still in equation 21, function root($\mathcal{M}$) obtains the variable of the root node of $\mathcal{M}$. Then, $\mathcal{M}_{\text{root}(\mathcal{M})}$ is the cofactor of $\mathcal{M}$ in respect to the variable of its root node. And $\mathcal{M}_{\overline{\text{root}(\mathcal{M})}}$ is the cofactor of $\mathcal{M}$ in respect to the negated variable of its root node.

To end, the activity function applied to BDDs with the same variable in their root nodes will be explained. First, the general *a*BDD structure is created for the signal of the root node (figure 5). The application of the activity function to some combinations of the cofactors of the operands will be attached to the four branches of the general structures.

It will be explained using the example of figure 15. The operand of the left is $\mathcal{M}$ and the operand of the right is $\mathcal{N}$. The four cofactors ($\mathcal{M}_A$, $\mathcal{M}_{\overline{A}}$, $\mathcal{N}_A$, $\mathcal{N}_{\overline{A}}$) in respect to the variable of the root node ($A$) have been drawn in the figure.

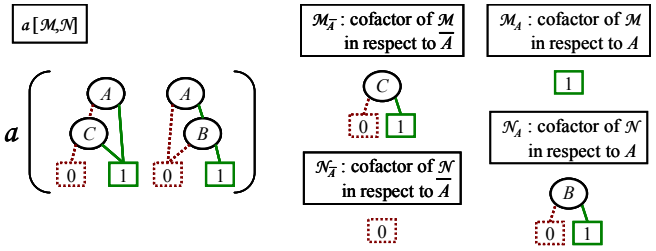


Fig. 15: Operands cofactors in respect to the variable of the root node

The application of the activity function to the four cofactor combinations has been drawn in figure 16. In the figure, the four combinations have been tagged. Tag 0→0 corresponds to $a[\mathcal{M}_{\overline{A}}, \mathcal{N}_{\overline{A}}]$, that is, the resulting BDD from the activity function applied to the cofactors in respect to the negated variable of the root node ($\overline{A}$). Tag 1→1 corresponds to $a[\mathcal{M}_A, \mathcal{N}_A]$, that is the resulting BDD from the activity function applied to the cofactors in respect to the variable of the root node ($A$). Tag 1→0 corresponds to $a[\mathcal{M}_A, \mathcal{N}_{\overline{A}}]$ and the tag 0→1 corresponds to $a[\mathcal{M}_{\overline{A}}, \mathcal{N}_A]$.

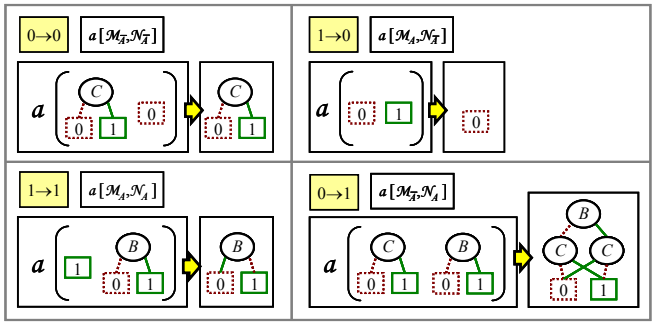


Fig. 16: Results of the activity function applied to the cofactors

These four BDDs will be attached to the four edges of the general *a*BDD structure created for variable $A$. The BDDs will be attached to the edges according to their tags. The BDD of tag 0→0 will be attached to the edge 0→0 and the BDD of tag 1→1 will be attached to the edge 1→1.

It is usual for the resulting BDDs of tags 0→1 and 1→0 to be equivalent. But it does not always happen, as in the example of figure 16. Since $P(x_{0\to1}) = P(x_{1\to0})$ and using equation 10, we have:

$$a_x = P(x_{0\to1}) + P(x_{1\to0}) = 2 \cdot P(x_{0\to1}) \quad (22)$$

Therefore, the arrangement can be done in a way that *a*BDDs can be simplified even more. If the smaller BDD is attached to the edge zero of the activity edges, smaller *a*BDDs are obtained. In this example, the BDD of tag 1→0 would be attached to the edge zero. Figure 17 shows this process: The BDD of tag 1→0 is smaller that the BDD of tag 0→1; therefore, BDD 1→0 is attached to the edge zero of the activity edges (those marked with $\frac{1}{2}a_A$). The resulting *a*BDD has been drawn on the right of figure 16.

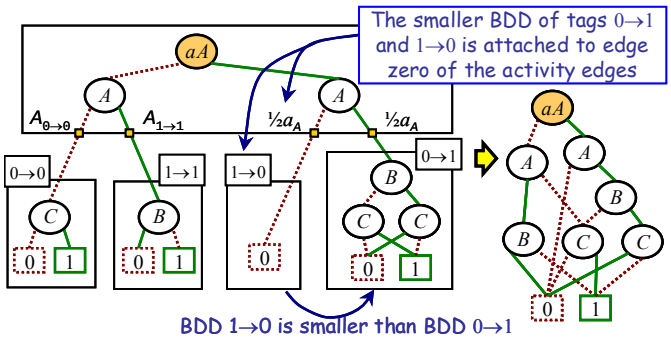


Fig. 17: Obtaining the activity BDD

To conclude this subsection, it will be remarked the importance of the activity operator definition in order to create *a*BDDs independently from TFBDDs, that is, just using probability BDDs. All the definitions and properties permit the univocal creation of *a*BDDs. It has also favored the process automation and the integration in a BDD manipulation computer program.

### *D. Comparison of the size of TFBDDs and aBDDs*

The size of the TFBDDs and *a*BDDs of some logic gates and multiplexers have been compared. The number of inputs has been incremented in order to obtain a general

rule. Table 1 shows the number of nodes and paths of logic functions for n-inputs gates. A multiplexer has been included in the last row, in which $n$ is the number of selection inputs; therefore, it has $2^n$ line inputs. For the multiplexer, the number of nodes follows an approximate sequence.

| Logic n inputs | number of nodes | | | % node reduction | number of paths | | | % path reduction |
|---|---|---|---|---|---|---|---|---|
| | BDD | TFBDD | *a*BDD | | BDD | TFBDD | *a*BDD | |
| and, or | n | 4n-1 | 3n-2 | $25\%\|_{n\to\infty}$ | n+1 | $n^2+2n+1$ | $\frac{1}{2}(n^2+3n)$ | $50\%\|_{n\to\infty}$ |
| xor | 2n-1 | 4n-1 | 2n-1 | $50\%\|_{n\to\infty}$ | $2^n$ | $2^{2n}$ | $2^n$ | $100\%\|_{n\to\infty}$ |
| mux | 2n-1 | $\sim 2(n+1)^2$ | $\sim(n+1)^2$ | $50\%\|_{n\to\infty}$ | 2n | $4n^2$ | $2n^2$ | $50\%\|_{n\to\infty}$ |

Table 1: Number of nodes and paths of the probability BDDs, TFBDDs and *a*BDDs for various logic functions.

AND, OR, NAND & NOR gates have the same number of nodes and paths. XOR & XNOR gates have the same number of nodes and paths. In Table 1, the columns called "BDD" indicate the number of nodes or paths of the probability BDD (the logic function BDD), which are smaller than those of the activity BDDs: TFBDD and *a*BDD. The column BDD has been included to show the differences between probability and activity BDDs. But the comparison has to be made between TFBDDs and *a*BDD, which are the ones used to compute signal activity. Table 1 shows that the usage of *a*BDDs instead of TFBDD lead to node reductions going from 25% to 50%. Greater reductions are achieved in the number of paths.

## IV. Experimental Results

The proposed activity operator has been integrated in the BuDDy BDD package [10], what has allowed to build an automated tool for analyzing the switching activity of VHDL combinational circuits. This tool analyzes the circuits and calculates the activity of the circuit signals. Just with the aim to compare the TFBDD and *a*BDD sizes, the tool elaborates both activity BDDs.

Many circuits have been analyzed at gate level and RTL. Some of the circuits have been modified changing their operand bus width. As a result, more than 200 analyses have been performed and in all the cases a reduction in BDD size has been achieved by the usage of *a*BDDs instead of TFBDDs.

Circuits have been obtained from references [14], [15]; besides many other academic circuits have been analyzed. The largest circuit is the 8051 ALU [15], being 10 bit width, having 759 logic gates and 40 inputs. The largest TFBDD had more than 1.2M nodes, its corresponding *a*BDD had 769K nodes (39% in node reduction).

The graphic of figure 18 shows the node reductions achieved in the experiments. The graphic X-axis has been ordered according to the node reduction achieved.

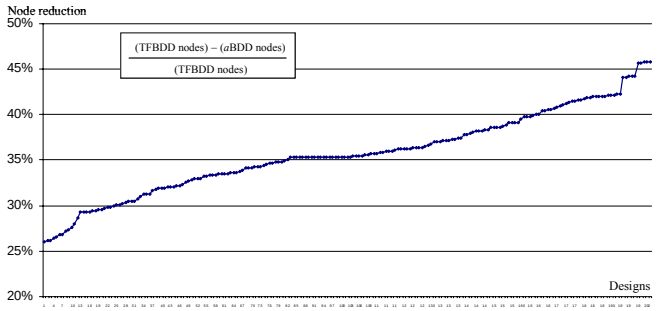


Fig. 18: Node reduction achieved using *a*BDDs instead of TFBDDs

The leftmost circuit of figure 18 is a RTL 15-bit adder/subtracter, in which the *a*BDD achieves a 26% reduction in the number of nodes. The rightmost circuit is a RTL ALU, in which the *a*BDD achieves a 46% reduction in the number of nodes. Since the proposed activity BDD does not imply any loss of accuracy, the activity results are identical independently of which activity BDD has been used (TFBDD or *a*BDD).

## V. Conclusions

A BDD representation to calculate the switching activity has been proposed. This activity BDD representation achieves significant reductions in the BDD size. An activity operator to build the proposed activity BDD from the logic function BDD has been defined. This operator definition allows its implementation in a BDD package and the construction of an automated tool.

Significant reductions in the activity BDD sizes have been achieved, going from 25% to 50% reduction in the number of nodes. Since the proposed activity BDD does not imply any accuracy loss, the activity results exact and equivalent to those used with the other representations.

This proposal is integrated in a wider research work aiming to reduce the complexity of the probabilistic activity estimation of digital designs.

## References


[1] International Roadmap for Semiconductors, 2007. http://public.itrs.net.

[2] R. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. on Computers*, vol. 35, no. 8, 1986.

[3] F. Machado, A. Abril, Y. Torroja, T. Riesgo, "An activity estimation tool for VHDL-RTL combinational circuits," *Design of Circuits and Integrated Circuits,* Barcelona, Spain, Nov. 2006

[4] M. Cirit, "Estimating dynamic power consumption of CMOS circuits," *Int. Conf. on Computer Aided Design*, Santa Clara, CA, USA, Nov. 1987.

[5] F. Najm, "Transition Density, A Stochastic Measure of Activity in Digital Circuits," *Design Automation Conf.*, San Francisco, CA, USA., June 1991

[6] A. Ghosh, S. Devadas, K. Keutzer, J. White, "Estimation of average switching activity in combinational and sequential circuits," *Design Automation Conf.*, Anaheim, CA, USA, June 1992

[7] P. Schneider, U. Schlichtmann, B. Wurth, "Fast power estimation of large circuits," *IEEE Design & Test of Computers*, Spring 1996

[8] R. Marculescu, D. Marculescu, M. Pedram, "Probabilistic modeling of dependencies during switching activity analysis," *IEEE Trans. on CAD of ICs and Systems*, vol.17, no. 2, Feb. 1998.

[9] H. Andersen, "An introduction to binary decision diagrams," *Lecture Notes for Efficient Algorithms and Programs,* Fall 1999. http://www.itu.dk/people/hra/bdd-eap.pdf

[10] J. Lind-Nielsen, "BuDDy: Binary Decision Diagram Package," 2002, http://buddy.sourceforge.net

[11] G. Theodoridis, S. Theoharis, D. Soudris, C. Goutis, "Switching activity estimation under real-gate delay using timed boolean functions," *IEE Proc. - Computers and Digital Techniques*, vol. 147, no. 6, Nov. 2000

[12] F. Machado, T. Riesgo, Y. Torroja, "A method for switching activity analysis of VHDL-RTL combinatorial circuits," *Int. Workshop on Power and Timing Modeling, Optimization and Simulation,* Montpellier, France, Sept. 2006.

[13] F. Machado, T. Riesgo, Y. Torroja, "Disjoint region partitioning for probabilistic switching activity estimation at register transfer level," *Int. Workshop on Power and Timing Modeling, Optimization and Simulation,* Lisbon, Portugal, Sept. 2008

[14] http://www.vhdl.org/rassp/vhdl/models/MSI/synth_models

[15] Oregano, http://www.oregano.at/ip/8051.htm, Oregano Systems